Digital Signal Processing

Mini-Project Report

Submitted by

Name : Bhavani Pediredla

Register Number : 220200052

E-mail ID : bhavani.p-26@scds.saiuniversity.edu.in

School of Study : CDS

Year of Study :2nd

Declaration

I hereby declare that the work presented in this mini-project report is my own. All sources used in this work have been properly cited and referenced.

<Signature>

<Name>

Date: _____

NOISE REDUCTION USING FILTERS AND DFT

1. Problem Statement

There is an ongoing issue with background noise during voice recording. The background noise primarily consists of unwanted frequencies, such as a persistent 200 Hz hum and occasional interference at specific frequencies. This noise is negatively impacting the quality of recorded voice content. To address this problem a comprehensive noise reduction approach is needed. The goal is to use a high pass filter and low pass filter to reduce background noise and by using DFT.

2. Motivation

The integration of noise reduction techniques specifically leveraging the power of discrete Fourier transform (DFT) and filters holds significant importance in applications ranging from audio processing and telecommunication to medical imaging. Unwanted noise is a pervasive challenge in these domains, capable of composing the integrity and clarity of desired signals by applying DFT and various filters we aim to elevate the quality of signals by mitigating or eliminating noise, thereby fostering a substantial improvement in data quality, communication reliability and overall user experience across diverse technologies landscape. This project seeks to address the critical needs for noise reduction, presenting a valuable contribution to the enhancement of signal fidelity and the optimization of various technological processes.

3. Methodology

What is Noise:

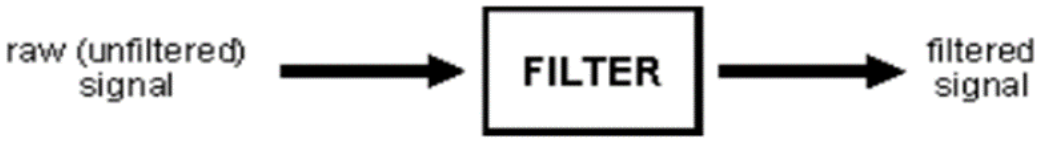
Noise is an unwanted sound that is unpleasant, loud, or disruptive. Other words

for noise include buzz, cacophony, commotion, crash, cry, explosion, roar, or turbulence.

INTRODUCTION TO FILTERS :

In signal processing, the function of a filter is to remove unwanted parts of the signal, such as random noise, or to extract useful parts of the signal, such as the components lying within a certain frequency range.

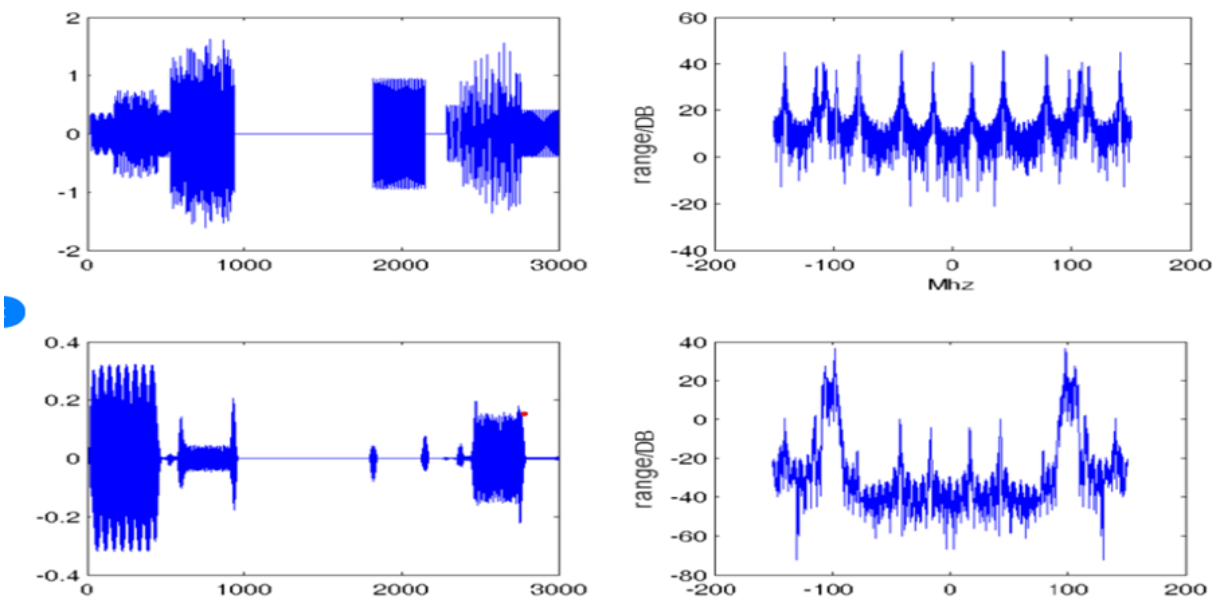
- *A filter is designed to selectively pass or reject signals based on their frequency content.*
- *Filters are commonly used to modify signals by emphasizing or attenuating specific frequency component*



Noise Reduction using Filters:

Noise reduction is the technique involved in extraction of the desired information from a signal.
The received signal is usually a mixture of information and noise. It is important to reduce the attenuation caused by random noise, since it produces the errors in information and lowers the quality of the signal. One of the most A challenging problem in digital signal processing is to extract the original information from a signal without any loss.

(a) Noise Signal before and After Filter



The filter is a device which shapes the signal waveform in a desired manner. The main purpose of filters in digital signal processing is to reduce the noise which improves the performance of the signal and to extract the desired information from the signal. There are many different
types of filters that are used to reduce the effects of noise. Like a low pass filter which takes the lower frequencies and rejects the higher frequencies, high pass filter , band pass filter and band stop filter .These types of filters have been proposed to improve the accuracy and efficiency of the information signal.

Why noise reduction using filters? How does it work?

Noise reduction using filters will
directly work on the time domain without explicitly transforming the signal into the frequency domain. But we can also transform signals from the time domain into the frequency domain by using transformation.
While more advanced signal processing techniques might involve
frequency domain transformation like Fourier Transform (DFT).

Types of Filters :

1. *Low-Pass Filter (LPF)*
2. *High-Pass Filter (HPF)*
3. *Band-Pass Filter (BPF)*
4. *Band-Reject Filter (BRF)*

Understanding of Low Pass Filter:

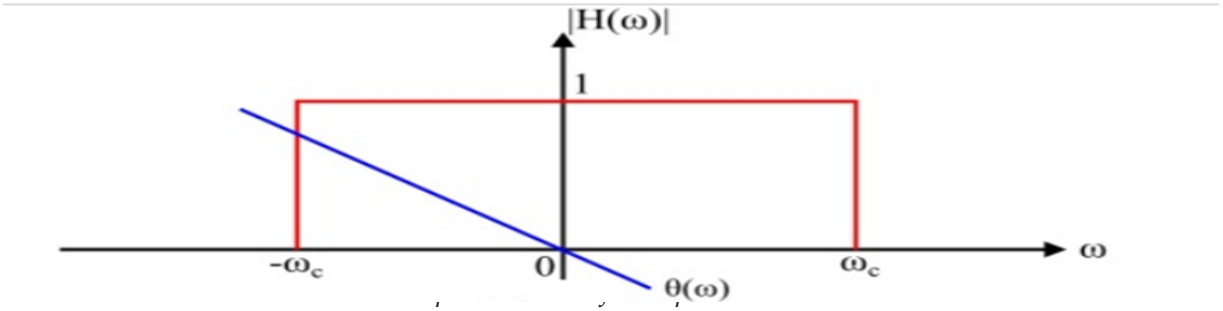
Electronic circuits known as low-pass filters selectively permit low-frequency signals to go through while attenuating or removing high-frequency ones. This makes them essential parts of many electronic devices, especially those that process audio and video, communicate across networks, have power supplies, or have control systems.

Their capacity to remove undesired high-frequency interference and noise guarantees these systems' correct operation and clarity.

They are employed to smooth out or restructure signals as well as eliminate high-frequency noise and interference. Low pass filters enhance the dependability and quality of electronic systems in this way.

Ideal Low Pass Filter. All frequencies below w_c are left unchanged and all amplitudes of the frequencies larger then w_c are set to zero:

$$H(j\omega) = \begin{cases} 1 & : \omega \leq \omega_c \\ 0 & : \text{elsewhere} \end{cases}$$

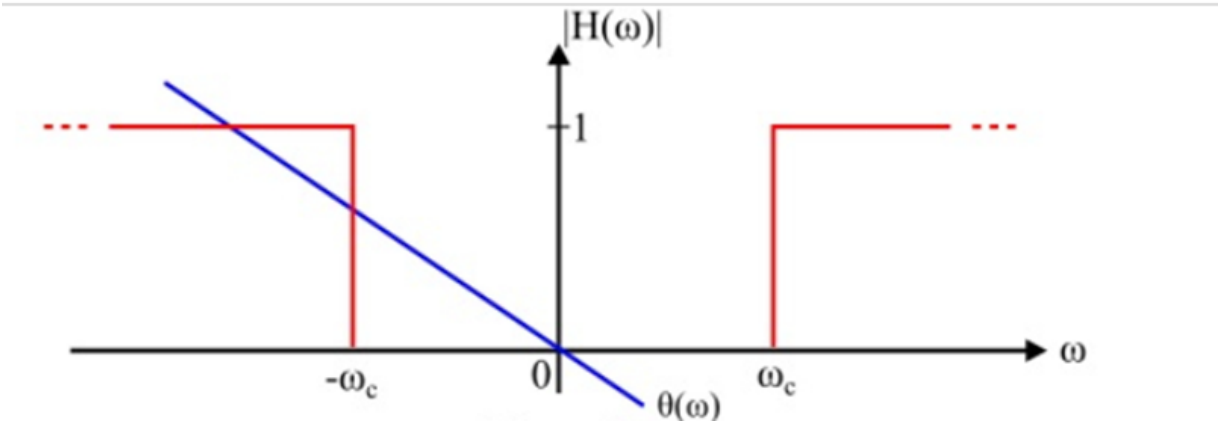


Understanding of High Pass Filter:

Signals below a cutoff frequency (the stopband) are attenuated by a high-pass filter, while signals above the cutoff frequency (the passband) are allowed. filter's design determines how much attenuation occurs. High-pass filters are frequently used to eliminate high-frequency interference, boost high-frequency signals to the right speakers in sound systems, eliminate low-frequency noise, and highlight high-frequency trends by eliminating low-frequency trends from time-series data.

Ideal High Pass Filter. A high pass filter is the opposite of the low pass filter:

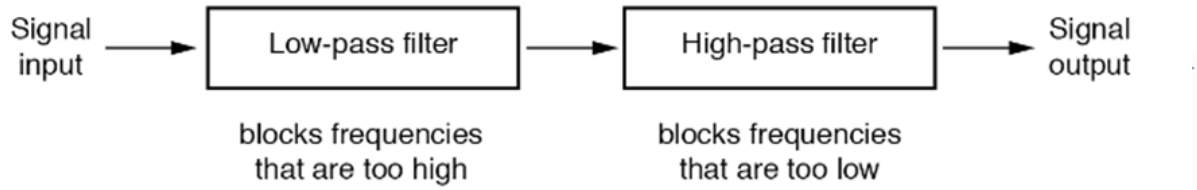
$$H(j\omega) = \begin{cases} 1 & : \omega > \omega_c \\ 0 & : \text{elsewhere} \end{cases}$$



Representation of high pass filter

Understanding of Band Pass Filter:

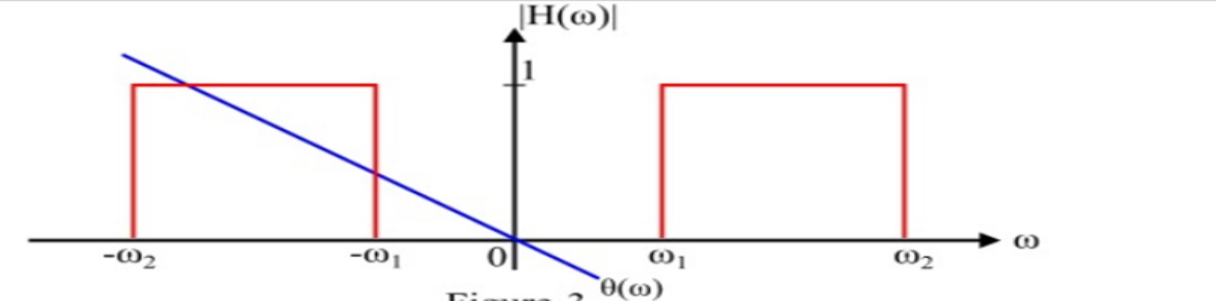
An electrical circuit or device known as a band pass filter attenuates or rejects frequencies outside of its frequency range while permitting only signals between predetermined frequencies to flow through. Although they are widely utilized in many fields of electronics, band pass filters are primarily used in wireless transmitters and receivers.



System level block diagram of a band-pass filter.

Ideal Band Pass Filter. The band pass filter passes the frequencies between ω_L and ω_H and suppresses all other frequencies:

$$H(j\omega) = \begin{cases} 1 & : \omega_L \leq \omega \leq \omega_H \\ 0 & : \text{elsewhere} \end{cases}$$



Representation of Band Pass filter

Understanding of Band Stop Filter:

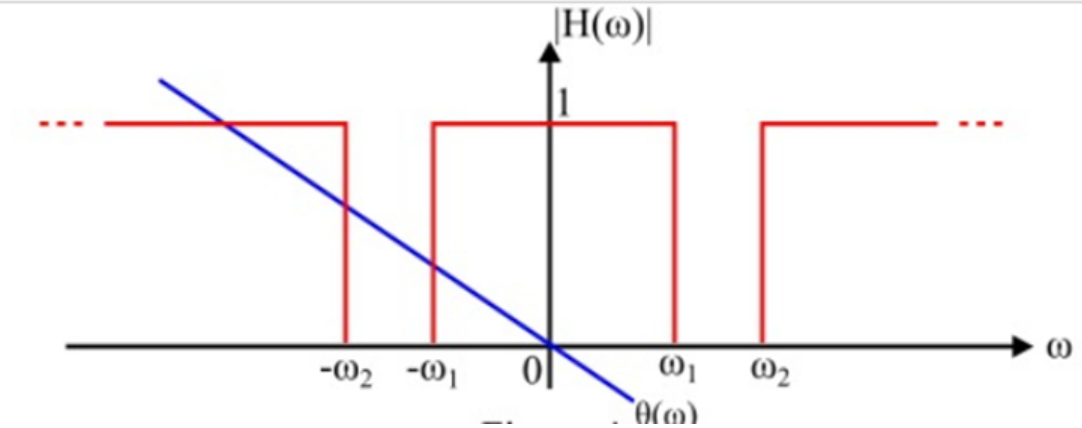
Electronic filters that attenuate a certain frequency band or range while permitting the remainder of the signal to pass through are called band stop filters, sometimes referred to as notch filters or band-rejection filters. They are frequently used to reduce noise or interference in audio, radio frequency, and telecommunications systems. They are made to remove undesired frequencies from a transmission.

Band stop filters are the reverse of bandpass filters in that they attenuate all other frequencies while allowing a particular range of frequencies to pass through. A band stop filter reduces or removes a specific frequency range from the output signal, whereas a bandpass filter chooses and amplifies that range.

A band pass filter is an electrical circuit or device that attenuates or rejects frequencies outside of its frequency range, allowing only signals between preset frequencies to pass through. Band pass filters are mostly employed in wireless transmitters and receivers, although being widely used in many other areas of electronics.

Ideal Band Reject Filter. The band reject filter is the opposite of the band pass filter:

$$H(j\omega) = \begin{cases} 0 & : \omega_L \leq \omega \leq \omega_H \\ 1 & : \text{elsewhere} \end{cases}$$



Representation of BandStop filter

An Introduction to the Discrete Fourier Transform :

Discrete Fourier Transform is referred to as DFT. The mathematical method of converting a signal from its original domain – typically the time domain – into the frequency domain is utilized in signal processing and digital signal analysis. For the purpose of examining and comprehending the frequency content of a discrete, sampled signal, the Discrete Fourier Transform is especially helpful.A class of these algorithms are called the Fast Fourier Transform (FFT).

The DTFT (discrete time Fourier transform) of any signal is X(ω)

$$X(\omega)= \sum_{n=-\infty}^{\infty} x[n]e^{-j\omega n}$$

$$X[n] = 1/2\pi \int_{-\pi}^{\pi} X(\omega)e^{j\omega n} d\omega$$

The DTFT has a significant flaw in that it cannot be computed by a computer because it requires an infinite-length summation.

The DFT assumes a finite length signal to handle this challenge.

N equations in N unknowns: if there are N samples in the time domain (x[n], 0 ≤ n ≤ N − 1), then there are only N independent samples in the frequency domain (X(ωk), 0 ≤ k ≤ N − 1).

First, assume that x[n] is nonzero only for 0 ≤ n ≤ N − 1. Then the DTFT can be computed as:

$$X(\omega) = \sum_{n=0}^{N-1} x[n]e^{-j\omega n}$$

Since there are only N samples in the time domain, there are also only N independent samples in the frequency domain:

$$X[k] = X(\omega_k) = \sum_{n=0}^{N-1} x[n]e^{-j\omega_k n} = \sum_{n=0}^{N-1} x[n]e^{-j^{2\pi kn/n}}$$

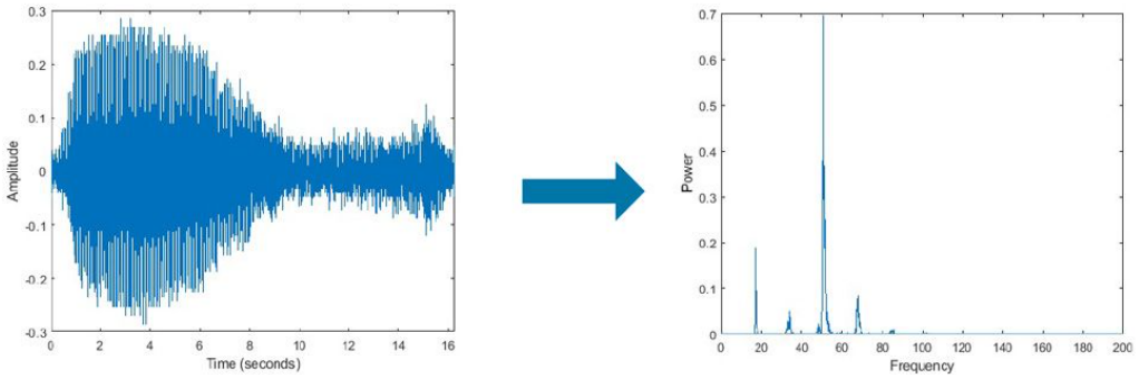
$$\omega_{k=}\ 2\pi k/N\ ,\ 0 \le k \le N - 1$$

Putting it all together, we get the formula for the DFT:

$$X[k] = \sum_{n=0}^{N-1} x[n]\, e^{-j2\pi kn/N}$$

What Is a Fast Fourier Transform (FFT)?

The discrete Fourier transform (DFT), which transforms discrete signals from the time domain to the frequency domain, is implemented in a highly optimized form by a fast Fourier transform (FFT). FFT calculations reveal details about the signal's phase, frequency content, and other characteristics.



4. Implementation :

Code 1:

1. *First set the parameters :*

- *Duration for 5 sec (That means the recording will only be allowed for 5 sec)*
- *Then the sampling frequency is set to 44100HZ (which is common sampling frequency . It is set like that because the human hearing range is roughly 20 hz to 20000 hz therefore a sampling frequency of 44100 HZ is more than sufficient to accurately capture and reproduce audio signal within the audible range.*

2. *Then the cutoff frequency is set to 2000 Hz (because we are using High Pass filter and want to attenuate low pass frequencies so we will be increasing the cutoff frequencies more than the passed frequency we can only observe how much we have to put cutoff frequency by observing the frequency domain of original signal of high frequency notes)*
3. *Then An audio recorder object (recObj) is created with a 16-bit resolution and a single channel indicates that an audio recording object named*

"recObj" is being created with specific parameters for recording audio data.

- *We are asking to display a message" telling Press any key to start recording " pause for some time till the person presses any key to start recording once the key is pressed "Start recording message is displayed " and 5 secs " end of the recording is displayed "*

4. *Getting the recorded data*
5. *Creating a subplot as (3,1,1) (The first number (3) indicates the number of rows of subplots in the figure. The second number (1) indicates the number of columns of subplots in the figure. And the third number (1) specifies the index of the subplot to be currently selected.)*
6. *plot((0:length(voiceData)-1)/fs, voiceData) (this is used to plot recorded audio signal)*
7. *Then applying high pass filter (we use high pass filter to remove low frequency noise from the corrupted noise signal)*
8. *Display the filtered voice signal*
9. *Play the original and filtered voice signal*
10. *Save the filtered voice signal*
11. *Apply FFT to the original signal so that it will give frequency domain of the original signal and also for filtered signal.*

Code 2:

1. *First set the parameters :*

- *Duration for 5 sec (That means the recording will only be allowed for 5 sec)*
- *Then the sampling frequency is set to 44100HZ (which is the common sampling frequency . It is set like that because the human hearing range is roughly 20 hz to 20000 hz therefore a sampling frequency of 44100 HZ is more than sufficient to accurately capture and reproduce audio signal within the audible range.*

2. *Then the cutoff frequency is set to 10 Hz (because we are using Low Pass filter and want to attenuate High pass frequencies so we will be decreasing the cutoff frequencies than the passed frequency we can only observe how much we have to put cutoff frequency by observing the frequency domain of original signal of high frequency notes)*
3. *Then An audio recorder object (recObj) is created with a 16-bit resolution and a single channel indicates that an audio recording object named "recObj" is being created with specific parameters for recording audio data.*
    - *We are asking to display a message" telling Press any key to start*

recording " pause for some time till the person presses any key to start recording once the key is pressed "Start recording message is displayed " and 5 secs " end of the recording is displayed "

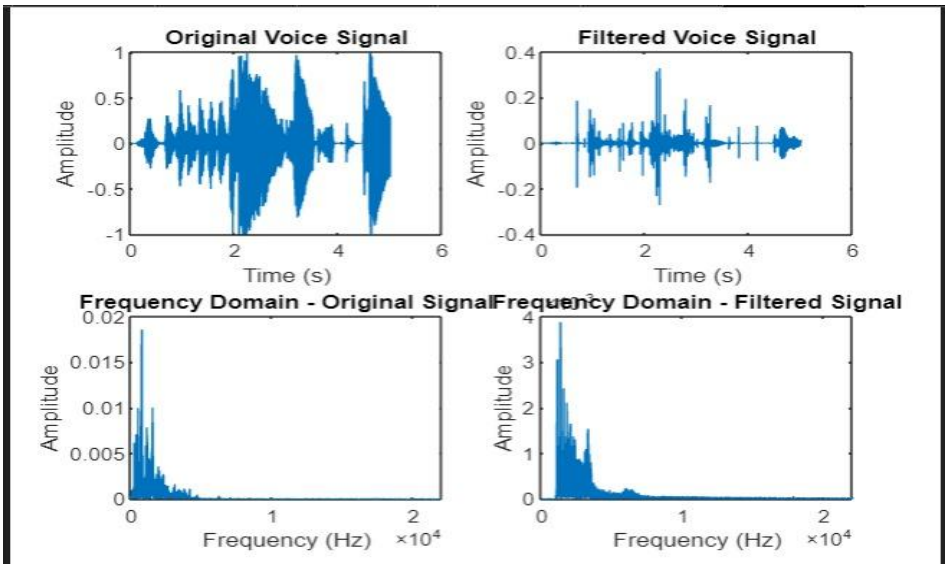
4. *Getting the recorded data*
5. *Creating a subplot as (3,1,1) (The first number (3) indicates the number of rows of subplots in the figure. The second number (1) indicates the number of columns of subplots in the figure. And the third number (1) specifies the index of the subplot to be currently selected.)*

6. *plot((0:length(voiceData)-1)/fs, voiceData) (this is used to plot recorded audio signal)*
7. *Then applying low pass filter (we use low pass filter to remove low frequency noise from the corrupted noise signal)*
8. *Display the filtered voice signal*
9. *Play the original and filtered voice signal*
10. *Save the filtered voice signal*
11. *Apply FFT to the original signal so that it will give frequency domain of the original signal and also for filtered signal.*

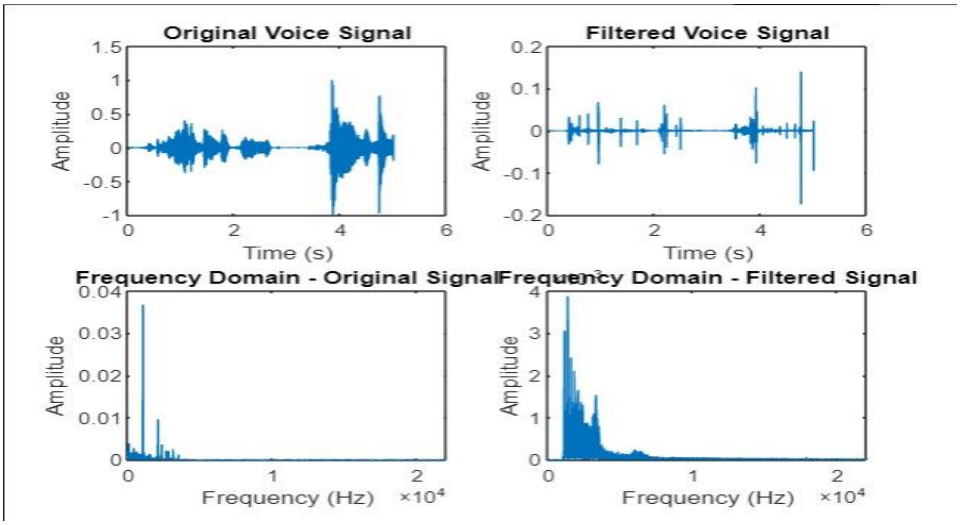
5. Simulation Results

Code 1:

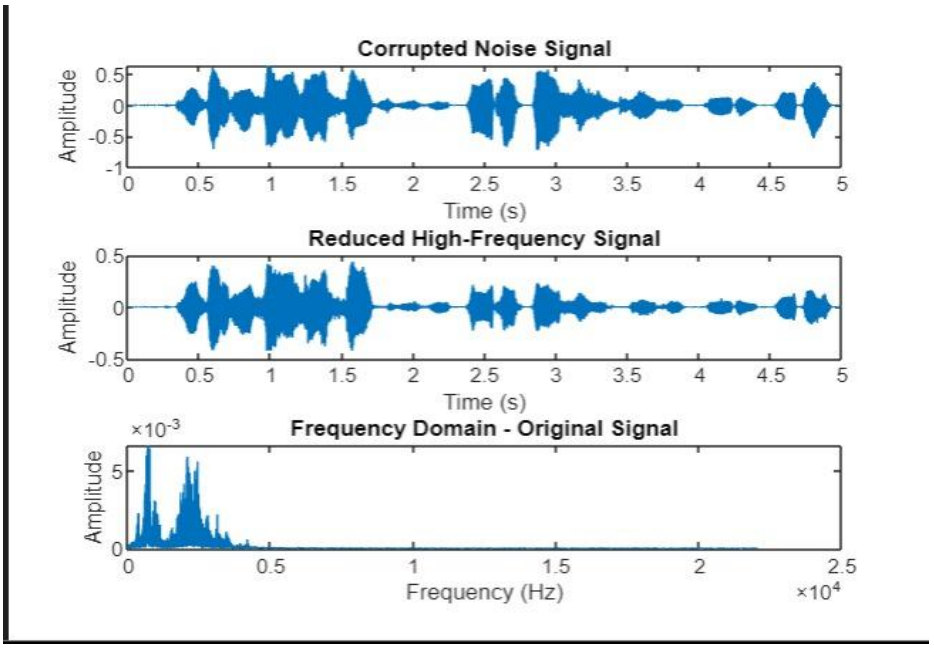
Test -1

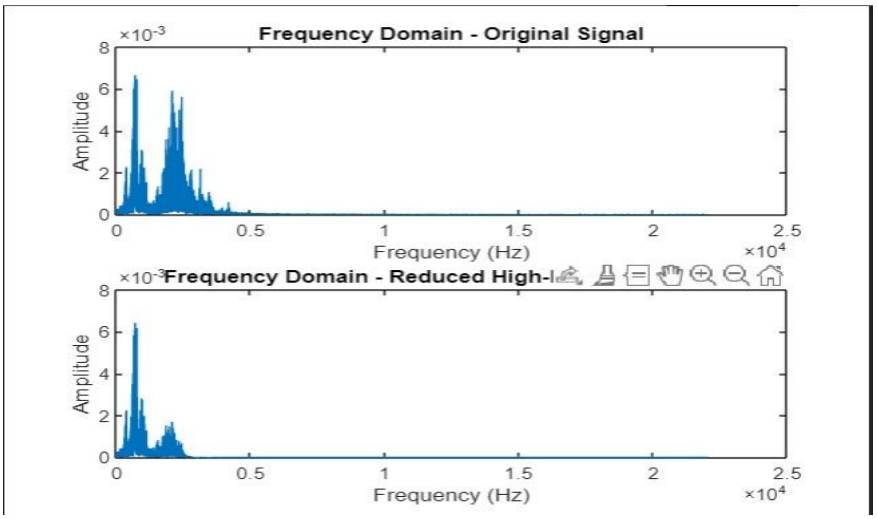


Test 2 :


Original Voice Signal
Filtered Voice Signal
Amplitude
Time (s)
Frequency Domain - Original Signal
Frequency Domain - Filtered Signal
Frequency (Hz)
×10^4

Code 2 :


Corrupted Noise Signal
Reduced High-Frequency Signal
Frequency Domain - Original Signal
Amplitude
Time (s)
×10^-3
Frequency (Hz)
×10^4



6. Conclusion

In summary, DFT is used in conjunction with high- and low-pass filters in the code to convert the original corrupted noise signal into the frequency domain and the filtered signal into the frequency domain. I've noticed that if we apply a high pass filter to a corrupted signal, the low frequency signal is attenuated, resulting in just high frequency signal output. Additionally, the opposite is true for low pass filters. As we approach the cutoff frequency, we must modify it while keeping an eye on the frequency domain frequencies. We will get the desired output by modifying the cutoff frequencies after several trials. I used filtered noise reduction to help in reducing high and low notes in music. I recorded a low note piano key while I was speaking, and I then applied a high pass filter to it to lower the low frequency signal. This means that the low note

piano signal will no longer be present in the original signal; instead, only the filtered signal containing the high frequency signal will be received. Then, using the high note piano key and talking as the original signal, I applied a low pass filter to it. This resulted in the removal of the high note frequency with the sole output being low frequency noise, which indicates that the high frequency piano key note was filtered away.

References

[1]"Characteristics of an Ideal Filter (LPF, HPF, BPF and BRF)," www.tutorialspoint.com. https://www.tutorialspoint.com/characteristics-of-an-ideal-filter-lpf-hpf-bpf-and-brf

[1]All About Circuits, "What Is a Low Pass Filter? A Tutorial on the Basics of Passive RC Filters," Allaboutcircuits.com, May 15, 2019. https://www.allaboutcircuits.com/technical-articles/low-pass-filter-tutorial-basis-passive-RC-filter/

[1]"What is Low Pass Filter," www.utmel.com. https://www.utmel.com/blog/categories/filters/what-is-low-pass-filter (accessed Nov. 19, 2023).

[1]"Definition of Low-Pass Filter | Analog Devices," Analog.com, 2023. https://www.analog.com/en/design-center/glossary/low-pass-filter.html

[1]Basar, "High Pass Filter: Definition, Circuit, Characteristics and applications," www.knowelectronic.com, Jul. 04, 2022. https://www.knowelectronic.com/high-pass-filter/

[1]"What is a Bandpass Filter? Definiton, design, response curve and applications of Bandpass Filter," Electronics Coach, Jan. 15, 2019. https://electronicscoach.com/bandpass-filter.html

[1]"Filter Design Gallery - MATLAB & Simulink Example," www.mathworks.com. https://www.mathworks.com/help/signal/ug/filter-design-gallery.html#zmw57dd0e1440 (accessed Nov. 19, 2023).

[1]D. Nowak and P. Schamid, "Introduction to Digital Filters," IEEE

Transactions on Electromagnetic Compatibility, vol. EMC-10, no. 2, pp. 210–220, Jun. 1968, doi: https://doi.org/10.1109/temc.1968.302947.

[1]"Readings | Digital Signal Processing | Supplemental Resources," MIT OpenCourseWare. https://ocw.mit.edu/courses/res-6-008-digital-signal-processing-spring-2011/pages/readings/ (accessed Nov. 19, 2023).

[1]L. Tan, Digital Signal Processing: Fundamentals and Applications. Elsevier, 2007. Accessed: Nov. 19, 2023. [Online]. Available: https://books.google.co.in/books/about/Digital_Signal_Processing.html?id=MMr7rP9TMOQC&redir_esc=y

Appendix:

Code : 1

```
% Set the parameters

duration = 5; % Recording duration in seconds

fs = 44100;   % Common Sampling frequency


% Adjust the cutoff frequency based on the background sound frequency

cutoffFrequency = 2000; % Set to a value higher than the background sound frequency


% Record voice

recObj = audiorecorder(fs, 16, 1);

disp('Press any key to start recording.');

pause;

disp('Start recording.');

recordblocking(recObj, duration);

disp('End of recording.');
```

```
% Get the recorded data
voiceData = getaudiodata(recObj);

% Display the original voice signal
figure;

subplot(3, 1, 1);
plot((0:length(voiceData)-1)/fs, voiceData);
title('Original Voice Signal');
xlabel('Time (s)');
ylabel('Amplitude');

% Apply a simple high-pass filter using 'highpass' function
voiceData_filtered = highpass(voiceData, cutoffFrequency, fs);

% Display the filtered voice signal
subplot(3, 1, 2);
plot((0:length(voiceData_filtered)-1)/fs, voiceData_filtered);
```

```
title('Filtered Voice Signal');

xlabel('Time (s)');

ylabel('Amplitude');


% Play the original and filtered voice signals

sound(voiceData, fs);

pause(duration + 1); % Pause to ensure the first sound is completed before the next

sound(voiceData_filtered, fs);


% Save the filtered voice signal (optional)

audiowrite('filtered_voice_highpass_background.wav', voiceData_filtered, fs);


% Apply FFT to the original signal

N_original = length(voiceData);

f_original = (0:N_original/2) * (fs/N_original);

Axk_original = 2 * abs(fft(voiceData) / N_original);

Axk_original(1) = Axk_original(1) / 2;
```

```
% Plot frequency domain of original signal
subplot(3, 1, 3);
plot(f_original, Axk_original(1:N_original/2+1));
xlabel('Frequency (Hz)');
ylabel('Amplitude');
title('Frequency Domain - Original Signal');

% Create a new figure for the frequency domain of the filtered signal
figure;
subplot(2, 1, 1);
plot(f_original, Axk_original(1:N_original/2+1));
title('Frequency Domain - Original Signal');
xlabel('Frequency (Hz)');
ylabel('Amplitude');

% Apply FFT to the filtered signal
N_filtered = length(voiceData_filtered);
```

```
f_filtered = (0:N_filtered/2) * (fs/N_filtered);
Axk_filtered = 2 * abs(fft(voiceData_filtered) / N_filtered);
Axk_filtered(1) = Axk_filtered(1) / 2;

% Plot frequency domain of filtered signal
subplot(2, 1, 2);
plot(f_filtered, Axk_filtered(1:N_filtered/2+1));
title('Frequency Domain - Filtered Signal');
xlabel('Frequency (Hz)');
ylabel('Amplitude');
```

Low pass: code 2 :

```
% Set the parameters

duration = 5; % Recording duration in seconds

fs = 44100;   % Common Sampling frequency


% Adjust the cutoff frequency based on the background sound frequency

cutoffFrequency = 10; % Set to a value higher than the background
sound frequency


% Record voice

recObj = audiorecorder(fs, 16, 1);

disp('Press any key to start recording.');

pause;

disp('Start recording.');

recordblocking(recObj, duration);

disp('End of recording.');


% Get the recorded data
```

```
corruptedNoise = getaudiodata(recObj);


% Display the original voice signal
figure;


subplot(3, 1, 1);
plot((0:length(corruptedNoise)-1)/fs, corruptedNoise);
title('Corrupted Noise Signal');
xlabel('Time (s)');
ylabel('Amplitude');


% Apply a low-pass filter using 'lowpass' function to reduce
high-frequency signals
lowFrequencySignal = lowpass(corruptedNoise, cutoffFrequency, fs);


% Display the low-frequency signal after reducing high-frequency
components
subplot(3, 1, 2);
plot((0:length(lowFrequencySignal)-1)/fs, lowFrequencySignal);
```

```
title('Reduced High-Frequency Signal');

xlabel('Time (s)');

ylabel('Amplitude');


% Play the original and filtered voice signals

sound(corruptedNoise, fs);

pause(duration + 1); % Pause to ensure the first sound is completed
before the next

sound(lowFrequencySignal, fs);


% Save the low-frequency signal (optional)

audiowrite('reduced_high_frequency_signal.wav', lowFrequencySignal,
fs);


% Apply FFT to the original signal

N_original = length(corruptedNoise);

f_original = (0:N_original/2) * (fs/N_original);

Axk_original = 2 * abs(fft(corruptedNoise) / N_original);

Axk_original(1) = Axk_original(1) / 2;
```

```
% Plot frequency domain of original signal
subplot(3, 1, 3);
plot(f_original, Axk_original(1:N_original/2+1));
xlabel('Frequency (Hz)');
ylabel('Amplitude');
title('Frequency Domain - Original Signal');

% Create a new figure for the frequency domain of the low-frequency signal
figure;
subplot(2, 1, 1);
plot(f_original, Axk_original(1:N_original/2+1));
title('Frequency Domain - Original Signal');
xlabel('Frequency (Hz)');
ylabel('Amplitude');

% Apply FFT to the low-frequency signal
```

```
N_lowFrequency = length(lowFrequencySignal);
f_lowFrequency = (0:N_lowFrequency/2) * (fs/N_lowFrequency);
Axk_lowFrequency = 2 * abs(fft(lowFrequencySignal) / N_lowFrequency);
Axk_lowFrequency(1) = Axk_lowFrequency(1) / 2;

% Plot frequency domain of low-frequency signal
subplot(2, 1, 2);
plot(f_lowFrequency, Axk_lowFrequency(1:N_lowFrequency/2+1));
title('Frequency Domain - Reduced High-Frequency Signal');
xlabel('Frequency (Hz)');
ylabel('Amplitude');
```

-END-